REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 10/23/95	**3. REPORT TYPE AND DATES COVERED** Proceedings, 25-26 September 1995

4. TITLE AND SUBTITLE A Portable Parallel Implementation of the U.S. Navy Layered Ocean Model	**5. FUNDING NUMBERS** N/A
6. AUTHOR(S) A.J. Wallcraft and D.R. Moore	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Planning Systems Inc. IC Dept. Math	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) SERDP 901 North Stuart St. Suite 303 Arlington, VA 22203	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** N/A

11. SUPPLEMENTARY NOTES
Presented at the Fourth International Parallel Computing Workshop (PCW), Imperial College, London, England, pp.1-20, 25-26 September 1995. No copyright is asserted in the United States under Title 17, U.S. code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Government purposes. All other rights are reserved by the copyright owner.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release: distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT *(Maximum 200 Words)*

FORTRAN was designed to run efficiently on primitive serial computers (it had to compete at birth head to head with machine assembly code) and its widespread use influenced the development of the most advanced machines used for numerical calculations in the 60's, 70's and 80's (c.f. CDC 64000, 6600, 7600 CRAY etc.). However, soon the performance limits achievable by a single serial processor economically became a limit to the monotonic growth of computer performance and alternative approaches (most typically dedicated multiprocessor configurations) have been offered as ways to surmount these constraints.

14. SUBJECT TERMS FORTRAN, serial computers, SERDP	**15. NUMBER OF PAGES** 20
	16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT unclass	**18. SECURITY CLASSIFICATION OF THIS PAGE** unclass	**19. SECURITY CLASSIFICATION OF ABSTRACT** unclass	**20. LIMITATION OF ABSTRACT** UL

(181-1995)

Modernising Legacy CFD Code:

A Portable Parallel Implementation of the US Navy Layered Ocean Model

by

A. J. Wallcraft, PhD (I.C. 1981)
Planning Systems Inc.

&

D. R. Moore, PhD (Camb. 1971)
IC Dept. Math.

The Problem:

FORTRAN has been in widespread use for ~40 years in scientific and industrial computation. On the order of 10^9 programmer-hours or 10^6 programmer-years has been using this language to create an enormously valuable legacy of code.

A major CFD Code (Computational Fluid Dynamics) typically represents 1 to 100 man-years of effort, and as such can be a significant financial asset if it can be kept running across generations of computers or it can become a major liability if it must be re-written when new computers become available.

FORTRAN was designed to run efficiently on primitive serial computers (it had to compete at birth head to head with machine assembly code) and its widespread use influenced the development of the most advanced machines used for numerical calculations in the 60's, 70's and 80's (c.f. CDC 6400,6600,7600 CRAY etc.).

However, soon the performance limits achievable by a single serial processor economically became a limit to the monotonic growth of computer performance and alternative approaches (most typically dedicated multiprocessor configurations) have been offered as ways to surmount these constraints.

Initially, access to gains in performance through multi-processor execution was possible only through use of a programming language unique to that machine. This cost constraint severely limited the exploitation of multi-processor machines and the runway for the take-off of their widespread use is littered with the bankrupt wrecks of such parallel computing companies as *Cray Computers, Connection Machines, Kendall Square, Hypercube,*

Floating Point Systems, Myrias, the makers of the DAP machine, etc.
However, in the last 10 years, two independent computing threads:

1. Inexpensive Desktops Scientific Work stations (e.g. SUN)

2. Cheap Personal Computers (IBM PCs & their clones)

have created a large successful commercial market selling very high performance serial computing units very cheaply.

[A 75 MHz Pentium© PC can be purchased for under $1,500 today and has a double precision FORTRAN performance of ~5 Megaflops on real CFD code. The current scientific workstations cost ~$20,000 and have a performance in the 50 Megaflop range on real code]

The attraction of harnessing this affordable computing power in multiple processor configurations has manifested itself in two hardware developments:

1. 'tightly bound' custom multiprocessor dedicated Massively Parallel Supercomputers using off the shelf scientific work-station CPU chips:

Chips	Machine
DEC Alpha	Cray T3D/E
SUN Sparc	Fujitsu AP1000
Intel 860	Paragon

2. 'loosely bound' clusters of 'off-the-shelf' workstations connected by:

slow	Normal Ethernet	10 Mbits/sec.
medium	Fast Ethernet	100 Mbits/sec.
fast	FDDI	>200 Mbits/sec.

Meanwhile, Cray Research, Inc. (who invented the supercomputer market and is still in business!) have produced a shared memory multiple CPU computer as its top performing model, connecting 4-64 500+ Megaflop vector processing CPU units to a large (~ 1 gigaword) memory. Each processor can see all of the memory.

On the software side, the inefficiency of a parochial language for each machine has been recognised as a problem. The proposed solution is a 'common-standard' of library subroutines to manage the exchange of data between separate processors in a uniform way from within FORTRAN. First PVM and more lately MPI (Message Passing Interface) have been created by enthusiastic committees of world experts. MPI is now available on most major UNIX platforms (including PCs via LINUX) and uses a standard set of FORTRAN Subroutine calls.

Thus we now have the prospect of producing a single version of a FORTRAN program that can run efficiently on serial, parallel or massively parallel computer architectures.

In an ideal world, all major code would be re-written from the ground up to exploit this portability. In practice, it is necessary to port mature programs over to this new computing environment.

I will now report on a successful case study of this porting process and what was learned from this effort that has wider implications of the FORTRAN CFD community.

The Navy Layered Ocean Model

This program models oceans as a stack ($3 \leq N \leq 6$) of shallow layers of variable thickness. The *shallow-water approximation.*

Each layer has a distinct density ρ_l, (*isopycnal*).

Within each layer the spherical *shallow water* equations are solved fo the horizontal velocities (u_i, v_i), layer thickness h_i, temperature T_i, salinity S_i and any tracer components C^j_i (pollutants, oxygen isotopes, experimental injectants, etc.).

The layers interact with each other through the pressure term which is a sum of the weight of the water column above each layer *i* at the point *(ϕ_i,θ_j) {longitude, latitude}* and any paramaterized mass exchange between the layers.

The uppermost layer is driven by surface winds, rain and sunshine from weather models, the deepest level experiences *bottom drag.* Each layer exerts drag on the levels above it and below it.

This model reflects decades of physical observations of the ocean which reveal them to be composed of distinct and recognisable water masses. The stacking into layers is caused by daily, seasonal and long term climatological processes. This stacking is found to be ver stable except in the extreme North Atlantic in winter when there are episodic *Deep Mixing events* converting surface water to bottom water a few ten cubic miles at a time. There are compensating *up-welling* events (El Niño, circum-Antarctic current) that occur sporat-ically around the world in known locations. This means that vertical mixing processes are unimportant over most of the worlds oceans.

Here are the actual equations that are modelled in each layer:

2.5 Summary-A Single Layer with Uniform Density

2.5.1 Continuity

$$\frac{\partial h}{\partial t} + \frac{1}{a\cos\theta}\left[\frac{\partial U}{\partial\phi} + \frac{\partial(V\cos\theta)}{\partial\theta}\right] = \text{sources - sinks} \tag{24}$$

2.5.2 Momentum

$$e_{\phi\phi} = \frac{\partial}{\partial\phi}\left(\frac{u}{\cos\theta}\right) - \cos\theta\frac{\partial}{\partial\theta}\left(\frac{v}{\cos\theta}\right), \tag{25a}$$

$$e_{\phi\theta} = \frac{\partial}{\partial\phi}\left(\frac{v}{\cos\theta}\right) + \cos\theta\frac{\partial}{\partial\theta}\left(\frac{u}{\cos\theta}\right). \tag{25b}$$

where the velocities (u, v) are defined from the transports (U, V) by:

$$hu = U, \tag{26a}$$

$$hv = V. \tag{26b}$$

$\vec{e_\phi}$ component:

$$\frac{\partial U}{\partial t} + \frac{1}{a\cos\theta}\left[\frac{\partial(Uu)}{\partial\phi} + \frac{\partial(Vu\cos\theta)}{\partial\theta} - Vu\sin\theta - a\Omega V\sin 2\theta\right] = -\frac{hg}{a\cos\theta}\frac{\partial h}{\partial\phi}$$

$$+hF_\phi + \frac{A_H}{a^2\cos^2\theta}\left[\frac{\partial(h\cos\theta\, e_{\phi\phi})}{\partial\phi} + \frac{\partial(h\cos^2\theta\, e_{\phi\theta})}{\partial\theta}\right] \tag{27}$$

$\vec{e_\theta}$ component:

$$\frac{\partial V}{\partial t} + \frac{1}{a\cos\theta}\left[\frac{\partial(Uv)}{\partial\phi} + \frac{\partial(Vv\cos\theta)}{\partial\theta} + Uu\sin\theta + a\Omega U\sin 2\theta\right] = -\frac{hg}{a}\frac{\partial h}{\partial\theta}$$

$$+hF_\theta + \frac{A_H}{a^2\cos^2\theta}\left[\frac{\partial(h\cos\theta\, e_{\phi\theta})}{\partial\phi} - \frac{\partial(h\cos^2\theta\, e_{\phi\phi})}{\partial\theta}\right] \tag{28}$$

2.5.3 Scalar transport

$$\frac{\partial(hT)}{\partial t} + \frac{1}{a\cos\theta}\left[\frac{\partial(UT)}{\partial\phi} + \frac{\partial(VT\cos\theta)}{\partial\theta}\right] =$$

$$+\frac{\kappa}{a^2\cos^2\theta}\left[\frac{\partial}{\partial\phi}\left(h\frac{\partial T}{\partial\phi}\right) + \cos\theta\frac{\partial}{\partial\theta}\left(h\cos\theta\frac{\partial T}{\partial\theta}\right)\right] + \text{sources - sinks.} \tag{29}$$

This approach is in contrast to *fixed-level* codes where the ocean is divided vertically into *j* fixed levels at depths z_i, $10 \leq i \leq 50$.

An ocean basin, such as the Pacific is covered with a uniform grid or mesh and finite difference equations approximating the partial differential equations set out for each layer are solved at each time step to advance the field variables in time.

The initial conditions are supplied from established climatological atlases and the ocean basins are *spun-up* for several years or decades to establish sensible motions, temperatures and salinity.

This model has been in research use since 1978 when its original authors published a study of the Loop Current in the Gulf of Mexico. A recent result from the model featured on the cover of *Nature* last summer (4 August 1994) and for the rest of that year and this year on their CD-ROM promotion literature.

This model is classic supercomputer legacy code, having started life on a Texas Instruments Advanced Scientific Computer and evolved through CRAY 1-S, CRAY X-MP, CYBER - 203, 205 and ETA 10, and i now running on an 8-processor CRAY C90.

In 1992 the US Department of Defense started a High Performance Computer Modernisation Program. This program is buying a *state-of the-art* computer for about $25,000,000 every year and establishing a Major Shared Resource Center to house and maintain this facility. These HPC centers will be the main resource for DoD sponsored computing for the foreseeable future.

There is very strong pressure to move all major DoD based computing onto these new machines, typically of MPP architecture.

Technical problems:

Although it is possible to solve the finite difference approximations t the *shallow water* equations explicitly (which lends itself immediately to a tiled approach on MPP architecture) the need to resolve the surface gravity wave mode*(tidal waves)* limit the possible time step t values too small(to ensure numerical stability) to be economical.

The semi-implicit approach filters out these surface waves at the cos of having to solve a Helmholz equation for each layer thickness variable, h_i.

It is necessary to solve this equation:

$$\nabla^2 h_i' - c_i^2 h_i' = s_i'(\phi,\theta)$$

over the true ocean basin shape and depth distribution.
This is the classic fractal problem. Here h_i' is a linear combination of the original h_i variables that allow a decomposition of the pressure driving terms in the velocity advection equations into separate modes. In general,

$$c_1 << c_2 \approx c_3 \ldots$$

where c_1 represents the whole ocean mode, while c_2, c_3 ... represent various internal modes of motion between layers.
The oceanographers call the c_1 mode *barotropic* and the c_2, c_3 ... modes the *baroclinic* modes.

For c_2, c_3 ... equation (1) can be solved to sufficient accuracy in a few SOR relaxations, a technique easily moved to MPP architecture, but for the c_1 case, SOR converges much too slowly and an exact solver must be used.

The serial computer code used the Capacitance Matrix Technique (CMT) of Hockney (1970). This approach can be set out schematicall as follows:

1. Given $s_1'(\phi,\theta)$, find h_1'', a solution field from an exact solver for a perfect rectangular domain.

2. Find an error field $e(\phi,\theta) = \nabla^2 h_1'' - c_1^2 h_1| - s_1'(\phi,\theta)$ at N special boundary points.

3. Calculate a correction field q' = [CMT] e where [CMT] is an N by N matrix.

4. Exact solve $\nabla^2 h_1' - c_1^2 h_1' = s_1'(\phi,\theta) - q'$ on the perfect domain to find a solution field with no error at the boundaries.

This CMT technique lends itself well to MPP architecture as the error calculation is a *local* process and the matrix multiply can be distributed efficiently over P processors.

The key is the exact solver!

Here the serial code used the Fourier Analysis - Cyclic Reduction technique of Temperton (1977).

This method uses a Discrete Fourier Transform in one space directio followed by a Tridiagonal equation solve for each Fourier mode in the orthogonal direction finished off by a Discrete Fourier Inverse to recover the physical variable values from their Fourier mode representation.

The efficiency of this method is so marked that it is in widespread use, but neither Fourier Analysis nor Tridiagonal equation solving liv well in an Parallel execution environment.

Our approach is to:

1. Adopt a 1-D mapping of long thin tiles to allow the FFTs to take place entirely within each tile.

2. Adopt a block - pipeline tridiagonal solver using the *Burn at Both Ends (BABE)* algorithm. This kept only a small number of processors idle at the start and the finish of the Tridiagonal solving cycle.

This strategy gave promise of being moderately efficient in a coarse MPP environment ($4 \leq P \leq 256$) .

Overall, the bulk of the ocean model calculation is done on a 2-D tilin covering the domain. Only for the baroclinic mode is the $s_1'(\phi,\theta)$ field mapped from its 2-D form to a 1-D form. This allows a simple SPMD (Single Program Multiple Data) approach to the rest of the code. Exchange of Halo Data is moved to a standard subroutine which is made architecture or communication protocol specific.

Only the Baroclinic Helmholz Solver requires specific communication modes beyond the standard 2-D tile Halo update.

We can analyse the cost of this approach within an MPP environment

Summarising the FACR(0) algorithm as we have implemented it:

1. Total Operation Count = $\frac{8 \log_2(N_x) N_x N_y}{P}$ *(to leading order)*

2. Total Data Movement: $= 2(P-1)\,N_x$

Values for N_x are 1024, 2048 or 4096. These give global resolutions o ¼°, 1/8 ° and 1/16° respectively.

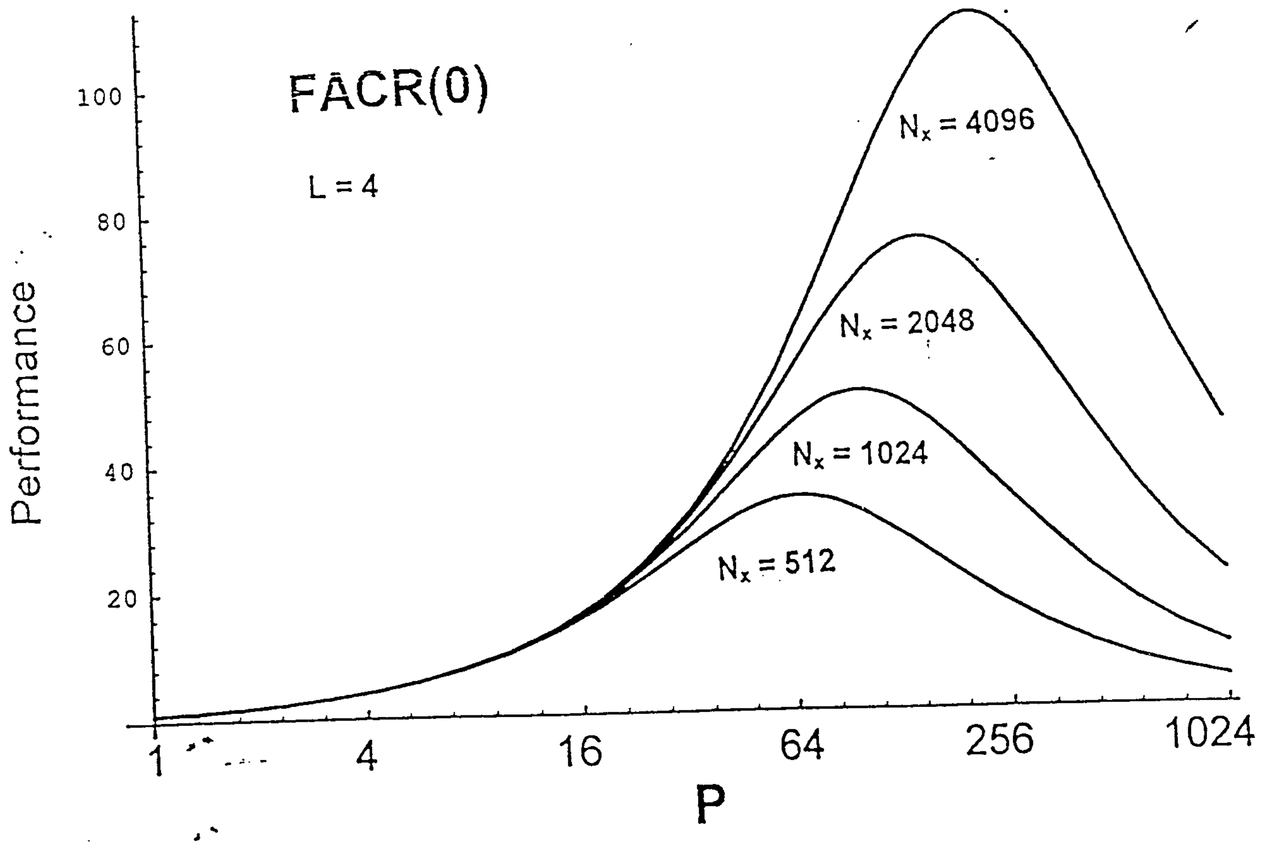
On a global grid where $\Delta\phi = \Delta\theta$, $N_y = N_x/2$.

If we adopt a naïve model of data communication, characterising it as being proportional to the time taken for a floating point operation wit a constant L, we can estimate the degree to which this approach scales over increasing numbers of processors P.

Let us characterise the performance of the algorithm as

$$\text{Performance}(p) = \frac{\text{time}(P=1)}{\text{time}(P)}$$

Here is a plot of this function for a value of 4 chosen for L.



This corresponds well to the Cray T3D timings we have obtained for 1024 x 768 case:

P	time	ratio
8	0.1298	
16	0.0666	1.95
32	0.0369	1.80
64	0.0227	1.63

The overall results suggests that this approach will allow the FACR(0 to run easily and efficiently for $32 \le P \le 256$ which should match wel the P values proposed for most general purpose MPP machines.

The efficiency of the complete CMT algorithm depends both upon the efficiency of the FACR(0) solver and on the number of special boundary points (B) that define the coastline of the ocean basin. As a coastline is the archtypical fractal length, B would be expected to scale on N_x in a non-linear fashion. Empirical results from the standard bathymetric data base suggests that $B \approx {}^1/_3 Nx^{3/2}$. In practice, the coastline is smoothed to limit the size of the CMT matrix.

For the current set of World Ocean models in use, we use:

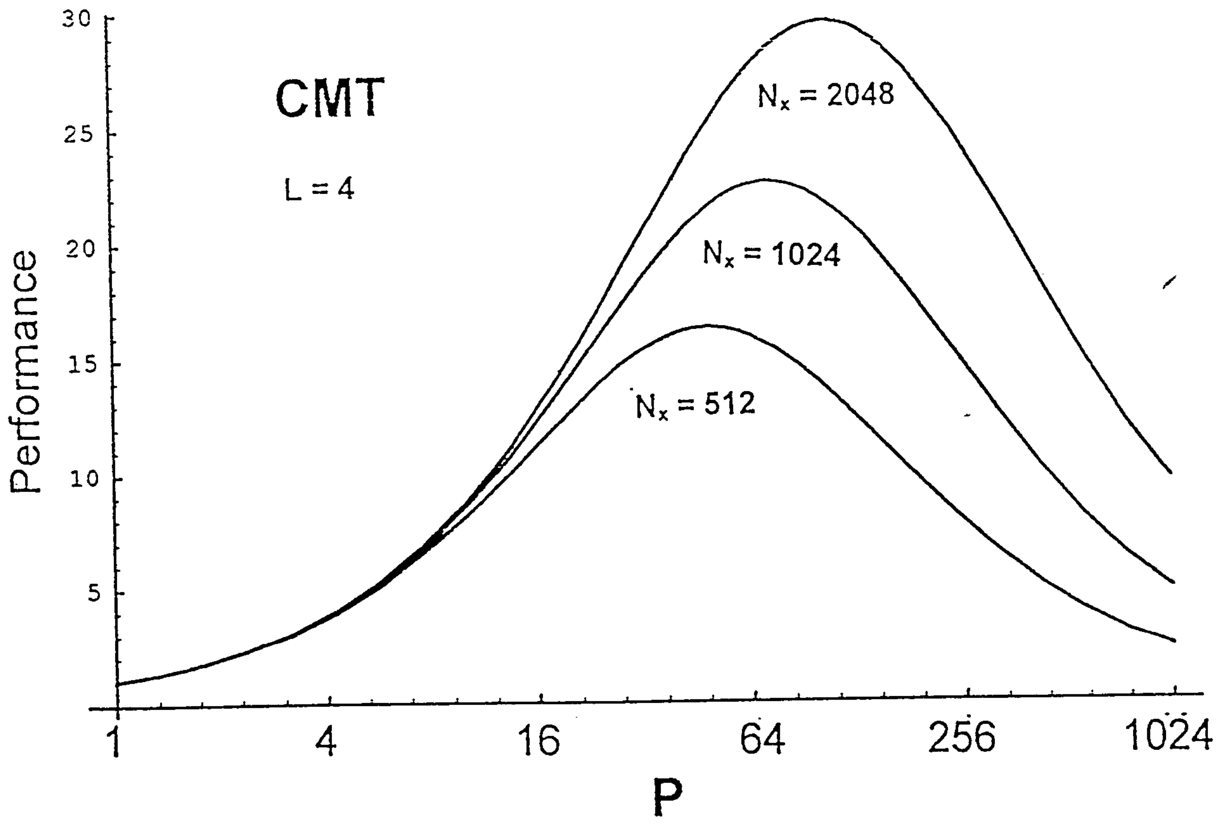
N_x	B	${}^1/_3 Nx^{3/2}$	Size of CMT (bytes)
512	3929	3861	62M
1024	8915	10922	318M
2048	18936	30843	1.4G
4096	???	87381	30.5G

For the complete CMT Algorithm (including the 2-D to 1-D and back mapping costs) we find:

Floating point ops per processor $= \dfrac{10\,B + 2\,B^2}{P} + 2\ \text{FACR}(0)$ costs

Data movement $= (1\text{-}1/P)(N_x^{\,2} + 2B) + 2\ \text{FACR}(0)$ costs

Here are the performance curves for an L value of 4 for the 3 cases where an actual value of B is set (there is strong pressure to smooth the boundary data to reduce the size of the capacitance matrix!):



Programming Style:

To achieve portability between distributed memory systems (Workstation Clusters) and shared memory systems (CRAY C90) a programming style was adopted which allowed for both models as well as for testing on a single serial machine on one tile.

The domain was divided up into M_{proc} by N_{proc} tiles such that:

$$M_{proc} \times N_{proc} = P.$$

M_{proc} and N_{proc} could both be 1.

A double set of outer loops over all the 2-D tiles was introduced around each computation statement: (e.g. for a Red-Black SOR type loop:

```
   COMMON/PROC2D/MPROC,NPROC
   DIMENSION R(NX,NY,MPROC,NPROC),
  +          P(NX,NY,MPROC,NPROC)

   DO JPROC=1,NPROC
     DO IPROC=1,MPROC
       DO J=2,NY-1
         DO I=2+MOD(J,2),NX-2+MOD(J,2)
           P(I,J,IPROC,JPROC) = ...
    ENDDO;ENDDO;ENDDO;ENDDO

   CALL PASS_HALO(P)
```

The outer loop limits `MPROC,NPROC` are set at compile time when possible to ensure efficient optimization. In advanced FORTRAN or in FORTRAN90/95 the two outer loop commands would be replaced by FORALL statements to indicate that simultaneous execution of th outer loops is possible (and encouraged!).

Here we see typical M_{proc} N_{proc} values used on the CRAY T3D (run in distributed memory mode) and the CRAY C90 (shared memory:

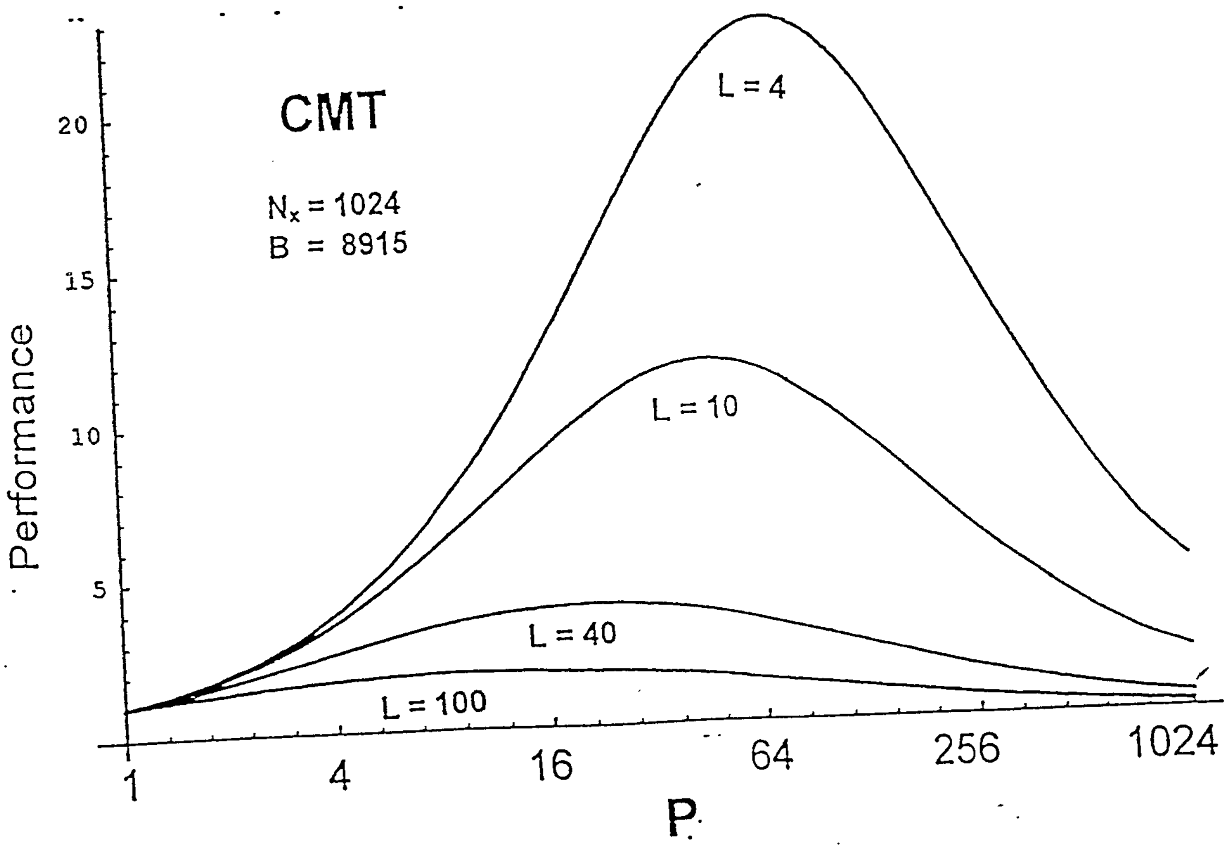
type	M_{PROC}	N_{PROC}
Shared Mem.	8	4
Distrib. Mem.	1	1

M_{PROC} and N_{PROC} can be set at Compile time or at Run time.

The Ocean Model code is now designed to port quickly to a variety of MPP ensembles. The default data communication protocol is MPI. Only one processor (#0) is allowed to do external I/O. The code has been tested on a cluster of SUN 10 work stations linked by ethernet running MPICH. Because the inter-tile communication is localized to a single generic Halo Update subroutine for the 2-D tile portion of the program (the bulk!), and to a few specialised communications in the CMT/FASCR(0) solver, tailoring the code to exploit a specific parochial data communication protocol (e.g. CRAY T3D in shared memory mode) is straightforward and can be implemented only wher critical.

The prospect of running the code on a cluster of High performance workstations is attractive. However, investigations of the efficiency o the CMT/FACR(0) algorithm when L is increased from 4 to the much higher values expected in a loosely bound cluster suggest that the scaling potential is much reduced.

Here are efficiency curves for the case:
$N_x = 1024$, $N_y = 576$, $B=8915$ and $L = 4, 10, 40, 100$.



Experience:

Dr. Wallcraft and I found it easiest to proceed in two steps:

1. Produce a shared memory version of the code using 2-D tiling (1-D tiling in the Barotropic Solver) isolating the inter-tile communication steps. This model will run on the CRAY C90 in multitasking mode.

2. Produce a MPI version of the code by writing MPI Subroutine calls to implement to inter-tile communication steps.

Step 1 ensured that the calculations on each tile could proceed independently and localised the communication.

Step 2 confined the MPI based code to a very few easily tagged subroutines that could be replaced by more efficient communication protocols when appropriate.

Debugging was possible on a small cluster (P=4) of SUN workstation running MPI. It was then possible to move the code directly onto a large P (64!) CRAY T3D with no further modification.

Local efficient serial codes can enhanced the overall performance considerably and the advantages of being able to exploit them may overcome the cost of reshuffling the data from a 2-D tile form to 1-D tile form and back again (the FFT routine!).

Conclusions for 2-D CFD Codes:

1. Producing MPP environment modern code from mature production serial CFD code need not be an impossible task.

2. Moving in stages from SPSD to SPMD to MPMD seems easier and less error prone than a single SPSD to MPMD jump.

3. Some classical serial algorithms will survive the move to coarse MPP environments ($P<100$) that are 'tightly bound' ($L<5$).

4. In large P environments, the size of L is critical.

5. Moderate sized workstation clusters ($P\sim8$) can be efficiently exploited to run these codes 5 times faster than a single workstation.

6. Artificial data structures to exploit local efficient serial code may be justified, especially for dedicated 'tightly bound' machines.

7. There may yet be life left in some 'old-dogs' [efficient serial algorithms] such as Hockney's Method in an MPP environment with $P \sim 100$.

Dr Wallcraft and I feel that we have been able to port a large mature program to a generic MPP environment with a simple standard of programming style (outer loops over tiles) and a default choice of inter processor communication (MPI). This creates a code that can be readily moved onto a new High Performance Computer and production type runs can be started with a minimum of porting effort. Later customisation to exploit local faster communications protocols can be exploited once MPI based experience has been gained that highlights specific data bottle necks. It is hoped that this leaves the NRL Ocean Modelling group well positioned to exploit the opportunities opening up from the DOD HPC initiative.